



TITLE:

微分差分作用素環用Grobner基底パッケージのO.D.E. section問題への応用(微分方程式の数式処理システムの研究)

AUTHOR(S):

高山, 信毅

CITATION:

高山, 信毅. 微分差分作用素環用Grobner基底パッケージのO.D.E. section問題への応用(微分方程式の数式処理システムの研究). 数理解析研究所講究録 1989, 681: 1-26

ISSUE DATE:

1989-02

URL:

<http://hdl.handle.net/2433/101153>

RIGHT:

微分差分作用素環用 Gröbner 基底パッケージの

O.D.E. section 問題への応用

徳島大学総合科学部 高山信毅
(Nobuki Takayama)

本稿で解法を紹介する問題 W の出発点となった問題は次の問題である。

問題 S。 多変数超幾何関数と Hypergeometric type O.D.E の関係を調べよ。

(多変数超幾何関数については [End1] を、Hypergeometric type O.D.E. については [Oku] を参照。)

二変数関数 $f(x, y)$ を (y を定数とみなし) x のみの関数とみたものを $f(x, y)$ の y -section と呼ぶことにする (resp. x を定数とみて y のみの関数とみたものを x -section)。たとえば、Appell の関数 F_1 の y -section, x -section は、Jordan-Pohhammer の O.D.E. をみたす関数である。また、 F_2, F_3 の section は、Goursat-Sasai の O.D.E. をみたす関数となっている。Jordan-Pohhammer の O.D.E. は 3 階の Hypergeometric system へ変換でき、Goursat-Sasai の O.D.E. は 4 階の Hypergeometric system へ変換できる。つまり、 F_1, F_2, F_3 は Okubo のアクセリ 1103x-70 の

Hypergeometric system の分類表 γ (Oku) で、たりおさまる γ である。
 では、他の多変数超幾何関数についてはどうであるか？
 というのが問題 S の由来である。(この問題については、[Suz-Ono]
 を参照。) こあるの例のような、よい関係が両者の間にある
 ば、多変数超幾何関数のモジュラ群の構造が Hypergeometric system を
 とおして、よく見えてくるわけである。

では問題 S を少し考えてみよう。 まず考えるべき関数を、
 Horn の表にありかつ確定特異点のみをもつ関数に限る([Erd1])、
 $F_1 \sim F_4$, $G_1 \sim G_3$, $H_1 \sim H_7$ のうち、独立変数の rational な変換で、 F_1, F_2
 へ reduce しないのは、 F_4, H_1, H_5 のみであるから([Erd2])、最も興
 味深いのは、 F_4, H_1, H_5 である。 まず F_4 の y -section を実験的に
 求めてみる。 実験であるから、 100×10^{-4} $\alpha, \beta, \gamma, \gamma'$ にも適当な値

$\alpha = 1/2$, $\beta = 1/5$, $\gamma = 9/7$, $\gamma' = -3/9$ を入れてやる。 §1 の算法
 I により、 y -section が求まる。 結果は fig 1。 ここで X_0 が x ,
 X_1 が y , D_0 が $\frac{\partial}{\partial x}$, $*$ は “かける” の意味である。 $(\frac{\partial}{\partial x})^4$ の係数
 が x の 5 次式であり、これは Hypergeometric type O.D.E. ではない。
 fig 1. の O.D.E. を L_0 とおき、fig 2. の 5 階の O.D.E. を L_1 とおく。
 L_1 は、§2 の算法 II により、計算した y -section である。 このと
 き、

$$L_1 = \left[\frac{1}{79597x_0 + 147223x_1 - 147223} (180x_0 \frac{\partial}{\partial x_0} + 173) \right] L_0$$

が成り立つ。 L_1 は Hypergeometric type O.D.E. であるが、 F_4 と L_1 は

77942000

$$\begin{aligned}
 & 396900*(X_0^2 - 2*X_0*X_1 - 2*X_0^2 + X_1^2 - 2*X_1 + 1)*(79597*X_0 + 147223*X_1 - \\
 & 147223)*D_0^4 *X_0^2 \\
 & - 1260*((625368263*X_1 + 804461513)*(X_1 - 1)*X_0 - 14*(11895489*X_1 - \\
 & 59949335)*X_0^2 - 212001120*(X_1 - 1)^3 - 246830297*X_0^3)*D_0^3 *X_0^3 \\
 & (735583815547*X_0^3 + 1143525132106*X_0^2 *X_1 - 2309462863673*X_0^2 - \\
 & 1433153630853*X_0^2 *X_1 - 165966988327*X_0 *X_1 + 1599120619180*X_0 + \\
 & 171720907200*X_1^3 - 515162721600*X_1^2 + 515162721600*X_1 - 171720907200)* \\
 & D_0^2 \\
 & + 7*(52460631566*X_0^2 + 132161838767*X_0 *X_1 - 164756093894*X_0 - 50053763133 \\
 & *X_1^2 - 22904798607*X_1 + 72958561740)*D_0 \\
 & + 97216*(79597*X_0 + 300403*X_1 - 300403)
 \end{aligned}$$

fig 1.

$$\begin{aligned}
 & - 71442000*(2*(X_1 + 1)*X_0 - (X_1 - 1)^2 - X_0^2)*D_0^5 *X_0^3 \\
 & - 56700*(29406*(X_1 + 1)*X_0 - 9491*(X_1 - 1)^2 - 19915*X_0^2)*D_0^4 *X_0^2 \\
 & - 180*(7*(4019626*X_1 + 4099223)*X_0 - 4724640*(X_1 - 1)^2 - 28625702*X_0^2) \\
 & *D_0^3 *X_0^3 \\
 & + (7419521543*X_0^2 - 3973785543*X_0 *X_1 - 4349562980*X_0 + 201787200*X_1^2 - \\
 & 403574400*X_1 + 201787200)*D_0^2 \\
 & + 7*(353788414*X_0 - 58817583*X_1 - 85732740)*D_0 \\
 & + 34317248
 \end{aligned}$$

fig 2.

よい関係にない。なぜなら、 L_1 は hypergeometric type ODE だが、
 \dots reducible であり、 F_4 のモジュラーの計算には、 L_1 は役に立たないから
 である。以上の観察より次のことがあつた。

“問題 S は、単に ODE section を求めるだけでは解けない。”

しかしながら、ODE section を求めるための算法をいろいろ
 考えておくことは、問題 S やその他の問題のいろいろな場面
 で有利である。このような動機により、本稿では、次の問
 題 W の解法を紹介することにする。

問題 W: (1) O.D.E. section 全体を求める算法を作れ。

(2) O.D.E. section 全体は単項イデアルとなるか、イデアルの生成
 元を求めるのがむづかしい場合は、生成元でなくとも、とにかく
 O.D.E. section をひとつ見つけよ。//問題

O.D.E. section を正確に定義しておく。

$$\mathcal{A} := \mathbb{C}(x, y) \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y} \right] \quad \text{微分作用素環}$$

\mathcal{A} の左イデアル \mathcal{I} が 0-次元イデアル であるとは、 \mathcal{A}/\mathcal{I} を
 $\mathbb{C}(x, y)$ 上の線型空間とみたとき、

$$\dim_{\mathbb{C}(x, y)} \mathcal{A}/\mathcal{I} < +\infty$$

となることである。

例。

$$L_1 = x \frac{\partial}{\partial x} (x \frac{\partial}{\partial x} + y \frac{\partial}{\partial y} + r - 1) - x (x \frac{\partial}{\partial x} + y \frac{\partial}{\partial y} + \alpha) (x \frac{\partial}{\partial x} + \beta)$$

$$L_2 = y \frac{\partial}{\partial y} (x \frac{\partial}{\partial x} + y \frac{\partial}{\partial y} + r - 1) - y (x \frac{\partial}{\partial x} + y \frac{\partial}{\partial y} + \alpha) (y \frac{\partial}{\partial y} + \beta')$$

とおく。 $L_1 f = L_2 f = 0$ の解は Appell の F_1 である。 L_1, L_2 により生成される \mathcal{A} のイデアルを \mathcal{J} とおく。このとき、

$$\dim_{\mathbb{C}(x,y)} \mathcal{A}/\mathcal{J} = 3$$

である。(パラメータ $\alpha, \beta, \gamma, \beta'$ には値が確定しているものとする) // 例

定義。 \mathcal{J} を \mathcal{A} の 0次元イデアルとする。このとき、 $\mathbb{C}(x,y) \left[\frac{\partial}{\partial x} \right]$ のイデアル

$$\mathcal{J} \cap \mathbb{C}(x,y) \left[\frac{\partial}{\partial x} \right]$$

を \mathcal{J} の O.D.E. y-sections のなすイデアルと呼ぶ。このイデアル

の要素を \mathcal{J} の O.D.E. y-section と呼ぶ。上のイデアルは単項イデアルとなるので、その生成元を、O.D.E. sections の生成元(generator)

と呼ぶ。// 注意

定理。 \mathcal{J} を \mathcal{A} の 0次元イデアルとする。 $\mathcal{J} \neq (0)$ なる、

$$\mathcal{J} \cap \mathbb{C}(x,y) \left[\frac{\partial}{\partial x} \right] \neq (0)$$

すなわち、自明でない O.D.E. section が必ず存在する。// 定理

注意。 例外的な場合を除き、

$$\dim_{\mathbb{C}(x,y)} \mathcal{A}/\mathcal{J} = \dim_{\mathbb{C}(x,y)} \mathbb{C}(x,y) \left[\frac{\partial}{\partial x} \right] / \mathcal{J} \cap \mathbb{C}(x,y) \left[\frac{\partial}{\partial x} \right]$$

となる。つまり、元の方程式系の rank と ODE section の generator の

rank は等しい。また、左辺 \geq 右辺 は、常に成り立つ。// 注意

(本稿は、二変数関数について述べているが、 n 変数の場合も、ほぼ同様に扱える。)

§1. 微分作用素環の Gröbner 基底による算法。

次のイデアルの Gröbner 基底を求める算法 (Buchberger 算法) を, lexicographic order で用いれば, O.D.E. section の generator は, Gröbner 基底の ひとつの要素として必ず求まる。

参考. [Fur] はこの算法 (多項式環) に対するチュートリアルである。A のイデアルや Weyl Algebra のイデアルの Gröbner 基底は, いろいろな問題に対して有効である。特殊関数の contiguity relation の導出と微分差分作用素環の Gröbner 基底一般については [Tak1] 及びその参考文献を見よ。Weyl algebra の Gröbner 基底, Hilbert 多項式の計算, cohomology の計算等については [Gel] を見よ。特殊関数を含む等式や二項係数の公式の自動証明システムについては, [Zei], [Tak2] を見よ。また, Gröbner 基底の概念は, ソリトン理論などでも用いられた [Nou]。// 参考。

算法 I は問題 w(1) の解である。(多項式環については, Buchberger のころからよく知られている算法)

算法 I. (本質的に, Buchberger による)

次のイデアルの Gröbner 基底を lexicographic order で求めよ。基底中に ODE sections の generator がある。// 算法

この算法の詳細については, 付録 I の REDUCE によるアルゴリズムの記述を見よ。付録 I は, この算法の解説兼プロトタイプ実装用の REDUCE 3.2, 3.3 用のプログラムである。

例. Horn の表の Φ_2 の微分方程式 ([Erd1]) の lexicographic order による

Gröbner 基底は,

$$\begin{cases} -(x-y)x \frac{\partial^3}{\partial x^3} - \left[(x-y)r + (y+2)x - x^2 - y + \beta x + \beta'y \right] \frac{\partial^2}{\partial x^2} \\ \quad - (\beta+1)(r-2x+y) \frac{\partial}{\partial x} + \beta(\beta+1), \\ y\beta \frac{\partial}{\partial y} - (x-y)x \frac{\partial^2}{\partial x^2} - [(r-x)(x-y) + \beta'y] \frac{\partial}{\partial x} + (x-y)\beta \end{cases}$$

ODE sections of generator

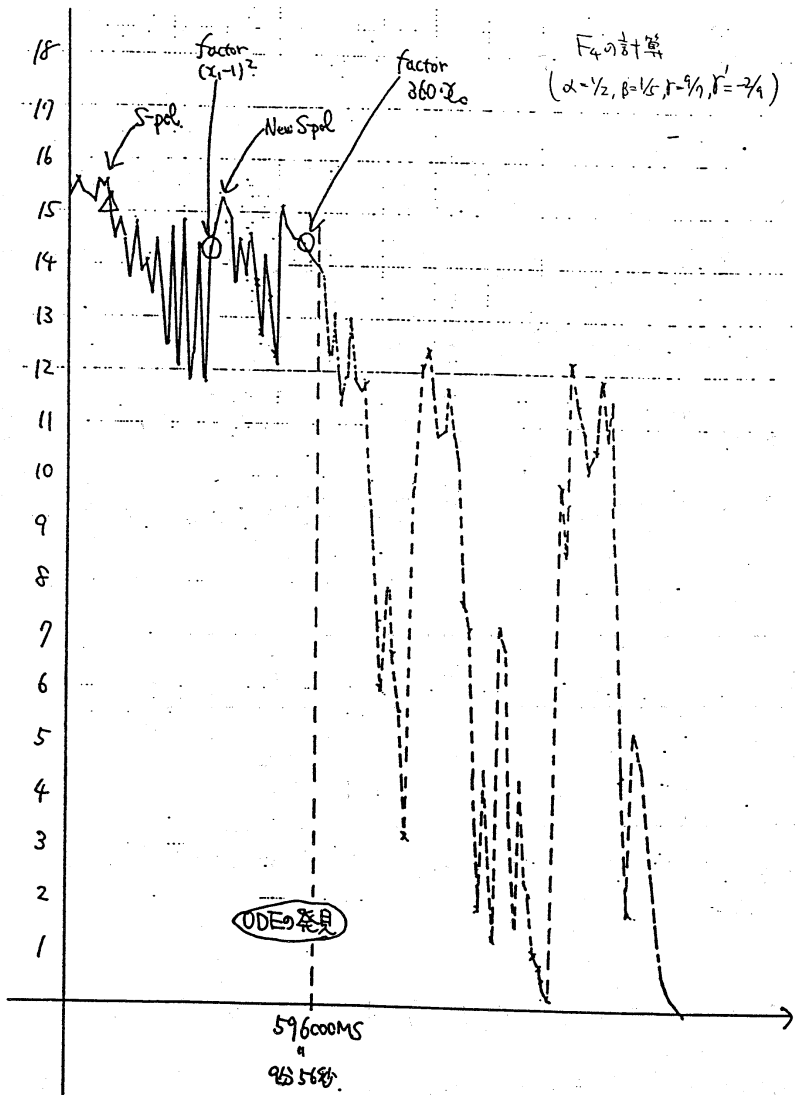


fig 3.

である。//例

この算法は、どんな 0-次元イデアルに対しても通用する、一般性が高い算法である。しかし、空間計算量が主あめで大きいのが欠点である。式の間膨張がおまてメモリ爆発がおきてしまう。fig3.は計算の進行にともなうて空きメモリ量がどう変化するかのグラフである。中間膨張の様子がわかる。

例.

関数名	F_1	F_2	F_3	F_4
ODE sectionが一般のパスに対して、算法Iで求まるか?	○	○	○	×

計算は、5 MByteのXインメモリをもつ VAXstation/II 上の、REDUCE3.3 でおこなった。//例

§2. 微分差分作用素環の Gröbner 基底を用いる算法

この節で述べる算法IIは問題W(2)の解である。ただし、多変数超幾何関数にしか通用しない。また、ODE sections の generator が必ず求まるとは限らない。一般に generator より高階になってしまう。(注。もし、一般のODEの解の continuity relation を求める算法が作れば、この算法は、一般の関数に対して通用する)。

算法IIの基本的 idea を述べる。Hypergeometric function を

$$F(x, y) = \sum_{k=0}^{\infty} f_k(y) C_k x^k$$

$f_k(y)$: Gauß の Hypergeometric function の族。

C_k : P-関数の積商。

と書く。 $f_k(y)C_k$ が k についてみたす差分方程式を求めて、
 $F(x, y)$ のみたす x についての O.D.E. を計算すれば、よみがえれるものである。

例.

$$F_2(x, y) = \sum_{k=0}^{\infty} \underbrace{F(x+k, y)}_{f_k(y)} \underbrace{\frac{(x, k)(\beta, k)}{(y, k)(1, k)}}_{C_k} x^k \quad // \text{例.}$$

さて、 f_k, C_k についての仮定より、 f_k, C_k は次のような関係式をみたすはかである。

$$\begin{cases} C_{k+1} = \frac{S_1(k)}{r_1(k)} C_k \\ C_{k+2} = \frac{S_2(k)}{r_2(k)} C_k \end{cases} \quad \text{ここで、 } S_i, r_i \in \mathbb{C}[k]$$

$$\begin{cases} H(k) f_k = f_{k+1} \\ L(k) f_k = 0 \end{cases} \quad \text{ここで、 } H, L \in \mathbb{C}(k, y) \left[\frac{\partial}{\partial y} \right].$$

以下、 E_k を k についてのシフト作用素

$$E_k f_k = f_{k+1}$$

とする。

算法 II.

入力. L, H, S_1, r_1, S_2, r_2

出力. $F(x, y)$ を x のみの関数とみたときの、 $F(x, y)$ がみたす O.D.E.

Step 1. $E_k - H(k), L(k)$ の Gröbner 基底を $\mathbb{C}(y, k) \left[\frac{\partial}{\partial y}, E_k \right]$ で、

lexicographic order $\frac{\partial}{\partial y} > E_k$ で求める。この基底は k についての 2 階の差分作用素を必ず含む (算法 I の類似)。

[もくには、 $f_k(y)$ は、高々超幾何関数 (Gaußの) だから公式集の公式より
適当に導く。しかしこれは3変数以上では通用しないので、
(3変数以上では)
Gröbner基底を用いる必要がある ([Tak1])]

この差分作用素を、

$$G_1 = a_2 E_k^2 + a_1 E_k + a_0$$

とおく。ここで、 $a_i \in \mathbb{C}(k, y)$ 。(注 $G_1 f_k = 0$ である。)

Step 2.

$$a_2 := a_2 * \frac{r_2}{s_2} * R; \quad a_1 := a_1 * \frac{r_1}{s_1} * R; \quad a_0 := a_0 * R;$$

(R は、 a_0, a_1, a_2 を多項式かつ $\gcd(a_0, a_1, a_2) = 1$ とするために適当にかける因子)

Step 3.

$$L := a_2 \Big|_{k=x\frac{\partial}{\partial x}-2} + x * \left(a_1 \Big|_{k=x\frac{\partial}{\partial x}-1} \right) + x^2 * \left(a_0 \Big|_{k=x\frac{\partial}{\partial x}} \right)$$

L を出力。 // 算法

例 Appell F_2 の場合.

$$\lambda \text{ 力: } H(k) = \frac{1}{\alpha+k} \left(y \frac{\partial}{\partial y} + 1 \right),$$

$$L(k) = y(1-y) \frac{\partial^2}{\partial y^2} + \left[y' - (\alpha+k+\beta+1)y \right] \frac{\partial}{\partial y} - (\alpha+k)\beta',$$

$$S_1(k) = (\alpha+k)(\beta+k),$$

$$r_1(k) = (r+k)(1+k),$$

$$S_2(k) = (\alpha+k)(\alpha+k+1)(\beta+k)(\beta+k+1),$$

$$r_2(k) = (r+k)(r+k+1)(1+k)(2+k).$$

$$1. \quad \begin{aligned} a_2 &= (k+\alpha+1)(y-1), \\ a_1 &= -\{(k+\alpha+1-\beta')y - 2(k+\alpha+1) + r'\}, \\ a_0 &= -(k+\alpha+1-r'). \end{aligned}$$

$$2. \quad \begin{aligned} R &= (\alpha+k)(\beta+k)(\beta+k+1) \quad \text{etc.} \\ a_2 &= (y-1)(r+k)(r+k+1)(1+k)(2+k), \\ a_1 &= -\{(k+\alpha+1-\beta')y - 2(k+\alpha+1) + r'\}(r+k)(1+k)(\beta+k+1), \\ a_0 &= -(k+\alpha+1-r')(\alpha+k)(\beta+k)(\beta+k+1). \end{aligned}$$

$$3. \quad \begin{aligned} L &= (y-1)(x\frac{\partial}{\partial x} - 2+r)(x\frac{\partial}{\partial x} - 2+r+1)(x\frac{\partial}{\partial x} - 2+1)(x\frac{\partial}{\partial x} - 2+2) \\ &\quad - x\{(x\frac{\partial}{\partial x} - 1+\alpha+1-\beta')y - 2(x\frac{\partial}{\partial x} - 1+\alpha+1) + r'\}(x\frac{\partial}{\partial x} - 1+r)x\frac{\partial}{\partial x}(x\frac{\partial}{\partial x} + \beta) \\ &\quad - x^2(x\frac{\partial}{\partial x} + \alpha+1-r')(x\frac{\partial}{\partial x} + \alpha)(x\frac{\partial}{\partial x} + \beta)(x\frac{\partial}{\partial x} + \beta+1) \end{aligned}$$

出力: L . //例.

算法ⅡをREDUCEでインポートし、Horn list [Erd1] の関数 $F_1 \sim F_4, G_1 \sim G_3, H_1 \sim H_7$ の O.D.E. y -section および $H_1 \sim H_7$ の O.D.E. x -section を計算した。 $F_1 \sim F_4, G_1 \sim G_3$ は x, y について対称なので、 x -section は計算しない。

計算結果。

使用計算機言語: VAX station II/rc (X1-X4 15 MByte)。REDUCE 3.3。

O.D.E. section のデータは約 800 Kbyte あるため、ここには掲載しない。section はすべて Hypergeometric type ODE (Fuchs 型かつ、最高次の微分の係数式の次数が、方程式のランクに等しい) である。

関数	方程式のランク	算法IIによるODE sectionのランク	特異点 (下の式=0 と ∞)
F_1	3	3	$xy(x-1)(y-1)(y-x)$
F_2	4	4	$xy(x-1)(y-1)(x+y-1)$
F_3	4	4	$xy(x-1)(y-1)(xy-x-y)$
F_4	4	5	$xy(x^2+y^2-2xy-2x-2y+1)$
G_1	3	4	$xy(x+y+1)(4xy-1)$
G_2	3	3	$xy(x+1)(y+1)(xy-1)$
G_3	4	4	$xy(2)x^2y^2-18xy-4x-4y-1)$
H_1	4	5	$xy(x-1)(y^2-4xy+2y+1)$
H_2	4	4	$xy(x-1)(y+1)(xy-y-1)$
H_3	3	6	$xy(4x-1)(y^2-y+x)$
H_4	4	5	$xy(4x-1)(y^2-2y-4x+1)$
H_5	4	6	$xy(27y^2x-36xy-y+16x^2+8x+1)$
H_6	3	5	$xy(4x+1)(y^2x-y-1)$
H_7	4	5	$xy(4x-1)(4xy^2-y^2-2y-1)$
H_1-x	4	5	
H_2-x	4	4	
H_3-x	3	3	
H_4-x	4	4	
H_5-x	4	5	
H_6-x	3	3	
H_7-x	4	4	

ここからの課題。

1. Xモリ爆発をおこさないような、Gröbner基底を求める算法を開発すること。
2. 微分作用素のかけ算の高速算法を開発すること。
3. 人間と対話的に処理をすすめやすいよう、数式をみやすいように、こづけて表示するための算法を開発すること。(何とおりか表示してくれるか、または、問題に即した意味理解をして表示してくれる。)

§. 付録1. $E(x_0, x_1, \dots) \left[\frac{\partial}{\partial x_0}, \frac{\partial}{\partial x_1}, \dots \right]$ 用 Gröbner 基底計算プログラム。

注. 解説版兼プロトタイプ版のため単純だが、計算速度がおとしい。
また REDUCE 3.2, 3.3 兼用のため冗長な部分がある。実用的パッケージを作るには、S-lisp を用い、かつ、微分作用素をどう計算機の中で表現するか(データ構造)まで、考えないといけない。cf. REDUCE 3.3 の groebner.red のソース。

注. このプログラムは REDUCE 3.3, 3.2 で動作する。disp() は、3.3 用, olddisp() は 3.2 用である。なお Frantz Lisp にのせている一部の REDUCE では動作しない。

0. Order は、 $\dots > \frac{\partial}{\partial x_2} > \frac{\partial}{\partial x_1} > \frac{\partial}{\partial x_0}$ なる lexicographic order を用いている。

1. !-VARS \wedge 変数 $\frac{\partial}{\partial x_0}, \frac{\partial}{\partial x_1}, \dots$ の個数を入力する。

例. $\frac{\partial}{\partial x_0}, \frac{\partial}{\partial x_1}$ のみを使用するのであれば、!-VARS := 2。

2. 対応 (大文字、小文字は区別しない)

$$\begin{array}{ll} x_0 \leftrightarrow x0 \leftarrow \text{zero} & \frac{\partial}{\partial x_0} \leftrightarrow D0 \\ x_1 \leftrightarrow x1 & \frac{\partial}{\partial x_1} \leftrightarrow D1 \\ \vdots & \vdots \end{array}$$

3. 微分作用素は、微分をうしひし \wedge も、てい、た正準形

$$\sum_{\alpha} a_{\alpha}(x) \left(\frac{\partial}{\partial x} \right)^{\alpha} \quad (\alpha \text{ は multi index.})$$

を2の対応で、多項式に直して入力する。

例. $(x_0 - x_1) \frac{\partial^2}{\partial x_0 \partial x_1} - a \frac{\partial}{\partial x_0} + b \frac{\partial}{\partial x_1} + C$ であらば、

$$(x0 - x1) * D0 * D1 - a * D0 + b * D1 + C$$

と入力。出力もこの形式である。(A は +, - について

は多項式環と同じ構造をしているため、多項式用 method を多く (関数)

利用している。) opmult(L, M) は作用素の積 LM を正準形にかえす。

4. プログラムのロジは、

in "dr0.rr"\$ in "db0.rr"\$

イテアルの generator を $ag(1), ag(2), \dots$ \wedge いる。 $ag(k)$ が最後のと王おりのマークとして、 $ag(k+1) := \emptyset$ と \emptyset を代入しておく。

5. O.D.E. section を求めるだけなら、

!-ODE := 1;

としておく。(答えは FFFF に入る。)

6. `gbasis()` で計算開始。答えも `ag` に入る。`disp()` は表示用。

7. 次は、Appell F_1 の section を求める計算例である。

($dh.r, rr \rightarrow dk.r$, $db.r, rr \rightarrow db.r$ とし、 c, a, b .)

入力: generator.

$$\begin{cases} x_0 \frac{\partial}{\partial x_0} (x_0 \frac{\partial}{\partial x_0} + x_1 \frac{\partial}{\partial x_1} + c - 1) - x_0 (x_0 \frac{\partial}{\partial x_0} + x_1 \frac{\partial}{\partial x_1} + a) (x_0 \frac{\partial}{\partial x_0} + b), \\ x_1 \frac{\partial}{\partial x_1} (x_0 \frac{\partial}{\partial x_0} + x_1 \frac{\partial}{\partial x_1} + c - 1) - x_1 (x_0 \frac{\partial}{\partial x_0} + x_1 \frac{\partial}{\partial x_1} + a) (x_1 \frac{\partial}{\partial x_1} + b') \end{cases}$$

700754721 は
lp と書ける。

出力:

$$\begin{aligned} & x_0(x_0-1)(x_1-x_0) \frac{\partial^3}{\partial x_0^3} + (ax_1x_0 - ax_0^2 + bx_1x_0 - 2bx_0^2 + bx_0 - cx_1 + cx_0 - x_1x_0b') \\ & + 3x_1x_0 + x_1b' - x_1 - 4x_0^2 + 2x_0) \frac{\partial^2}{\partial x_0^2} + (b+1)(ax_1 - 2ax_0 - bx_0 + c - x_1b' + x_1 - 2x_0) \frac{\partial}{\partial x_0} \\ & - b(b+1)a \end{aligned}$$

% reduce
REDUCE 3.3, 15-Jan-88 ...

← reduce の起動.

1: `on comp;`

← 以後、読み書き procedure はここに記す。

2: `in "dh.r" $ in "db.r" $`

← $\mathbb{Q}(x,y)[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}]$ 用. β -basis Package (total order 用) の読み込み。
(α 用)

-VARS := 2

506514, length 2268 bytes

GBASIS プログラム読み込み終了。

4: `f1:=opmult(x0*d0,x0*d0+x1*d1+c-1)-x0*opmult(x0*d0+x1*d1+a,x0*d0+b);`

F1 := - X0*(A*B + A*X0*D0 + B*X1*D1 + B*X0*D0 - C*D0 + X1*X0*D0*D1 -

$$X1*D0*D1 + X0^2 * D0^2 - X0^2 * D0^2 + X0^2 * D0^2)$$

5: `ff1:=opmult(x1*d1,x0*d0+x1*d1+c-1)-x1*opmult(x0*d0+x1*d1+a,x1*d1+bp);`

FF1 := - X1*(A*X1*D1 + A*BP - C*D1 + X1^2 * D1^2 + X1*X0*D0*D1 - X1^2 * D1^2

$$+ X1*D1*BP + X1*D1 - X0*D0*D1 + X0*D0*BP)$$

6: `ag(1):=f1$ ag(2):=ff1$ ag(3):=0;`

AG(3) := 0

← `gbasis()` 用の変数設定。

F_1 の方程式を出力。
`opmult` は operator の積を
この procedure.

10: !-ode:=1; ← basis を計算するためにはなく、
 ODE section において、時点で停止する必要がある。

Time: 68 ms

11: gbasis(); ← $\{ag(1), ag(2), ag(3)\}$ の
 gbasis の計算。

Size of old Ag is 0

Compacted. Size of reduced Ag is 2

***** ID fill no longer supported --- use lists instead

*** Z3 Z2 Z1 Z0 are non zero

Reduced Field factor: $X1 \cdot X0$

Spol of 1 and 2

***** ID fill no longer supported --- use lists instead

*** Z2 Z1 Z0 are non zero

Reduced Field factor: $-(A - C + 1)$

1: New s-pol is added. $A \cdot B \cdot X1^2 - A \cdot B \cdot X0^2 + A \cdot X1 \cdot X0 \cdot D0 - A \cdot X0^2 \cdot D0 + B \cdot$

$X1^2 \cdot D1 + B \cdot X1 \cdot X0 \cdot D0 - B \cdot X1 \cdot D1 - B \cdot X0^2 \cdot D0 - C \cdot X1 \cdot D0 + C \cdot X0^2 \cdot D0 + X1 \cdot X0^2$

$\cdot D0 - X1 \cdot X0^2 \cdot D0 - X1 \cdot X0 \cdot D0 \cdot BP + X1 \cdot X0 \cdot D0 + X1 \cdot D0 \cdot BP - X0^3 \cdot D0^2 + X0^2 \cdot$

$D0^2 - X0^2 \cdot D0$

Size of old Ag is 2

***** ID fill no longer supported --- use lists instead

*** Z3 Z2 Z1 Z0 are non zero

Reduced Field factor: $X0 \cdot (X0 - 1)$

Found target! Answer is FFFF.

Completion is completed. ← 答えは FFFF に。

Basis consists of 3 elements. (Ag(1), ...)

Time: 242386 ms

← 時間。約 242 秒 = 約 4 分。

12: disp(ffff)s

← ffff の表示。

D0

[X0]

[X0 - 1]

[X1 - X0]

結果。

以下略。

//7. 計算例。

/usr/users/z007/basis/Rims/dh0.rr

1989/1/14 14:8:25 1 p. 1--

%Order definition for the ring of differential Operators.

% dh0.rr 1988/02/26

% 1989/01/14

% Ring of differential operators with meromorphic coefficients.

% Order is lexico graphic. (....D3>D2>D1>D0)

%

% This program is implemented on REDUCE 3.2 and 3.3.

% We do not use ``symbolic procedures`` to make a simple program.

% But the time performance is not good for the reason.

% Modify here %%%%%%%%%%

% !-VARS:=2;

% %%%%%%%%%%

% %%%%%%%%%% global vars %%%%%%%%%%

% X0,X1,... D0,D1,D2,... are global variables.

% %%%%%%%%%%

% off factor;

symbolic operator numberp;

symbolic operator xx;

symbolic operator dd;

symbolic operator myzz;

symbolic procedure xx k;

intern compress(list('x,car explode(k)))\$

symbolic procedure dd k;

intern compress(list('d,car explode(k)))\$

symbolic procedure myzz k;

intern compress(list('z,car explode(k)))\$

% %%%%%%%%%%

procedure head(f);

begin

scalar m;

for m:=(!-VARS-1) step (-1) until 0 do f:=mylterm(f,dd(m));

return f;

ends

% headcof(f) is the head coefficients of f.

procedure headcof(f);

%for monic. Calculate head coefficients

begin

scalar m;

for m:=(!-VARS-1) step (-1) until 0 do f:=mylcof(f,dd(m));

return f;

ends

procedure mylcof(f,v);

begin

scalar tmpDen;

tmpDen:=den(f); f:=num(f);

if (deg(f,v) neq 0) then return(lcof(f,v)/tmpDen)

else return (f/tmpDen);

ends

procedure mylterm(f,v);

/usr/users/z007/basis/Rims/dh0.rr

1989/1/14 14:8:25 2 p. 57--

```

begin
    scalar tmpDen;
    tmpDen:=den(f); f:=num(f);
    if deg(f,v)=0 then return (f/tmpDen)
    else return( lterm(f,v)/tmpDen );
ends$
procedure mypart(f);
begin
    for i:=(!-VARS-1) step (-1) until 0 do f:=mylterm(f,dd(i));
    return f;
ends$

% f and g must be monomials.
% If f>g then 1 else 0.
procedure larger(f,g);
begin
    scalar fd,gd;
    if f neq 0 then fd:=f/headcof(f)
    else fd:=0;
    if g neq 0 then gd:=g/headcof(g)
    else gf:=0;
    return( largers0(fd,gd,!-VARS-1) );
ends$
procedure largers0(f,g,m);
begin
    if m < 0 then return 0;
    if deg(f,dd(m))>deg(g,dd(m)) then return 1;
    if deg(f,dd(m))=deg(g,dd(m)) then return (largers0(f,g,m-1));
    return 0;
ends$

% If f is ODE of x0 then 1 else 0.
procedure isODE(f);
begin
    if f = 0 then return 0;
    if larger(dl,head(f))=1 then return 1 else return 0;
ends$

% f and g must be monomials.
% If f is m-reducible by g then 1 else 0.
procedure reducible(f,g);
begin
    if infield(den(f/g))=1 then return 1 else return 0;
ends$

% If f does not contain D0,D1,... then 1 else 0.
procedure infield(f);
begin
    scalar result,m;
    if numberp(f) then return 1;
    result:=1; m:=!-VARS-1;
    while (m>=0) and (result=1) do <<
        if deg(f,dd(m)) neq 0 then result:=0;
        m:=m-1;
    >>;
    return result;

```

/usr/users/z007/basis/Rims/dh0.rr

1989/1/14 14:8:25 3 p. 113--

```

ends$
% binomial coefficient.
procedure binom(a,m);
begin
  scalar result,k;
  if (a<=0) or (m<=0) or (a=m) then return 1;
  if 2*m>a then m:=a-m;
  result:=1;
  for k:=1:m do <<
    result:=result*a/k;
    a:=a-1;
  >>;
  return result;

```

ends\$

```

% Function for pretty printing
%%%%%%%%%% disp() for REDUCE 3.3 %%%%%%%%%%%
procedure disp(f);
begin
  scalar tmp;
  if (f=0) then << write "0"; return(0); >>;
  while f neq 0 do <<
    tmp:=head(f); f:=f-tmp;
    disp!-monomial(tmp);
  >>;

```

ends\$

```

procedure disp!-monomial(f);
begin
  scalar tmp,tmp0,m,k;
  tmp:=1;
  for m:=0:(!-VARS-1) do <<
    k:=deg(f,dd(m));
    f:=f/(dd(m)**k);
    tmp:=(dd(m)**k)*tmp;
  >>;
  write tmp;
  tmp:=factorize(f);
  for each tmp0 in tmp do <<
    write " [" ,tmp0,"]"$
  >>;
  write "-----"$

```

end \$

```

%%%%%%%%%% disp() for REDUCE 3.2 %%%%%%%%%%%
procedure olddisp(f);
begin
  scalar tmp;
  write "/"-----";
  while f neq 0 do <<
    tmp:=mypart(f);
    on factor; write " ",tmp; off factor;
    f:=f-tmp;
    tmp:=0; %to save memory.
  >>;

```

/usr/users/z007/basis/Rims/dh0.rr

1989/1/14 14:8:25 4 p. 169--

```

write "-----/"
ends$

% opmult(f,g) is the multiplication of f and g as operators.
procedure opmult(f,g); % don't write tail recursive program to save memory.
begin
  scalar tmp,result;
  if (f=0) or (g=0) then return 0;
  result:=0;
  while ( f neq 0 ) do <<
    tmp:=mypart(f);
    result:=opmultsl(tmp,g)+result;
    f:=f-tmp;
  >>;
  return result;
ends$

% f must be monomial.
procedure opmultsl(f,g);
begin
  scalar a,fd,k,m,alpha,tmp;
  if numberp(g) or numberp(f) then return (f*g);
  a:=headcof(f);
  fd:=f/a;
  if numberp(fd) then return (f*g);
  for k:=0:(!-VARS-1) do <<
    tmp:=0;
    alpha:=deg(fd,dd(k));
    for m:=0:alpha do <<
      tmp:=tmp+binom(alpha,m)*df(g,xx(k),m)*dd(k)**(alpha-m);
    >>;
    g:=tmp;
  >>;
  return (a*g);
ends$

% opPower(f,p) is f*f*....*f ( p times ).
procedure opPower(f,p);
begin
  scalar tmp;
  tmp := 1;
  if p = 0 then return 1;
  if p = 1 then return f;
  for i:=1:p do <<
    tmp :=opmult(f,tmp);
  >>;
  return(tmp);
ends$

on gcd;
write " !!!!! ON GCD. !!!!! ";
ends$

```

/usr/users/z007/basis/Rims/db0.rr

1989/1/14 14:52:2 1 p. 1--

```

% The generic procedures to compute the Groebner basis
% for Differential Operators, difference operators and Weyl algebras.
% An order must be defined in dh0.rr
%      db0.rr 1988/12/05
%      1989/01/14
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% modify here %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SIZEAG:=40; SIZEAS:=80;
!-ODE := 0;      % If you want to stop when you find ODE in the
                  % Ag, then !-ODE:=1 else !-ODE:=0.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
array Ag(SIZEAG);      % G-basis

clear FFFF;      % If you want to find ODE in the generators, the result
                  % is put to FFFF. cf. !-ODE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% variables for the system. %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
!-DBG := 0$      % Debug flag of mred().
!-DBG0 := 0$      % Debug flag of reduceAg()
!-DBG1 := 0$      % Debug flag of gbasis(). Output the candidate of S-pol.
!-RED1FLAG := 0$ % used in mred1(). 1==> reduced.
!-REDFLAG := 0$ % used in mred().
!-FFFFFLAG:=0$ % Flag for the ODE check
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% end of definition of global vars %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
symbolic operator myzz;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% modify here %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
procedure checkAgForStop();
begin
    scalar result;
    result:=0;
    return( result );
ends

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% mred1(f,g) is m-reduction of f by g.
procedure mred1(f,g);
begin
    scalar hf,hg,result;
    !-RED1FLAG := 0;
    if (g=0) then write "Error in mred1() : devide by 0";
    if numberp(g)=1 then return 0;
    result:=f;
    repeat <<
        f:=result;
        if (f=0) then result:=0
        else <<
            hf:=head(f);
            hg:=head(g);
            if (reducible(hf,hg) = 0) then
                result := f
            else <<
                !-RED1FLAG:=1;
                result:=f-opmult(den(hf/hg)*hf/hg,g)/den(hf/hg);
                result:=result*den(result); % (x+y)/2;
            >>;
        >>;
    >>;
end

```

/usr/users/z007/basis/Rims/db0.rr

1989/1/14 14:52:2 2 p. 57--

```

    >>;
    >> until ( f = result );
    return result;
ends$

% mredAg(f) is m-reduction of f by the set Ag.
procedure mredAg(f);
begin
    scalar oldf,ii,tmp,k,ftmp,mm,kk;
    !-REDFLAG:=0;
    k:=0; oldf := 0;
    while (oldf neq f) and (f neq 0) do <<
        ii:=1; oldf:=f;
        while (Ag(ii) neq 0) and (f neq 0) do <<
            tmp := mred1(f,Ag(ii));
            if !-RED1FLAG=1 then !-REDFLAG:=1;
            if ( !-DBG = 1 ) then <<
                write "###mredAg():",f," ==> ",tmp;
                write "                by Ag(",ii,")  :  ",Ag(ii);
            >>;
            if (tmp neq f) then <<
                k:=k+1;
                f:=tmp; tmp:=0; %to save mem.
            >>;
            ii:=ii+1;
        >>;
    >>;
    return(f );
ends$

% mredaa(f) is m-reduction of f by the set aa.
procedure mredaa(f);
begin
    scalar oldf,ii,tmp,k,ftmp,mm,kk;
    !-REDFLAG:=0;
    k:=0; oldf := 0;
    while (oldf neq f) and (f neq 0) do <<
        ii:=1; oldf:=f;
        while (aa(ii) neq 0) and (f neq 0) do <<
            tmp := mred1(f,aa(ii));
            if !-RED1FLAG=1 then !-REDFLAG:=1;
            if ( !-DBG = 1 ) then <<
                write "###mredaa():",f," ==> ",tmp;
                write "                by aa(",ii,")  :  ",aa(ii);
            >>;
            if (tmp neq f) then <<
                k:=k+1;
                f:=tmp; tmp:=0; %to save mem.
            >>;
            ii:=ii+1;
        >>;
    >>;
    return(f );
ends$

% sp(f,g) is the s-polynomial of f and g.

```

/usr/users/z007/basis/Rims/db0.rr

1989/1/14 14:52:2 3 p. 113--

```

procedure sp(f,g);
begin
  scalar hf,hg,lc,tmp,hcoff,hcofg,coflc,ftmp,kk,mm;
  if (f=0) then return g;
  if (g=0) then return f;
  hf:=head(f); hg:=head(g);
  hcoff:=headcof(hf); hcofg:=headcof(hg);
  hf:=hf/hcoff; hg:=hg/hcofg;
  lc:=hf*hg/gcd(hf,hg);
  coflc:=hcoff*hcofg/gcd(hcoff,hcofg);
  tmp:=opmult((coflc/hcoff)*lc/hf,f)-opmult((coflc/hcofg)*lc/hg,g);
  tmp:=tmp*den(tmp);
  return( tmp );

```

ends

```

procedure getsizeAg();
begin
  scalar n;
  n:=1;
  while Ag(n) neq 0 do n:=n+1;
  return(n-1);

```

ends

```

procedure getsizeAs();
begin
  scalar n;
  n:=1;
  while As(n) neq 0 do n:=n+1;
  return(n-1);

```

ends

% The procedure reduces the set Ag.

```

procedure reduceAg;
begin
  scalar n,j,k,tmp,m;
  n:=getsizeAg();
  array aa(n),newAg(n+1);n:=getsizeAg();
  for k:=1:n do newAg(k):=Ag(k);
  repeat <<
    % newAg ==> (compaction) ==> Ag
    n:=getsizeAg(); newAg(n+1):=0;
    k:=1;
    for j:=1:(n+1) do <<
      Ag(k):=newAg(j);
      if (Ag(k) neq 0) then k:=k+1;
    >>; % compaction
    Ag(k):=0; n:=getsizeAg();
    % Ag ==> newAg
    for j:=1:(n+1) do newAg(j):=Ag(j);
    if !-DBG0 neq 0 then write "#reduceAg() : size of Ag is ",n;
    % Do reduction of newAg
    K:=0;
    while (K < N) do <<
      K:=K+1;
      % aa== complement of newAg(k) except 0.
      m:=1;

```


/usr/users/z007/basis/Rims/db0.rr

1989/1/14 14:52:2 4 p. 169--

```

    for j:=1:n do <<
        if (j neq k) and (newAg(j) neq 0) then <<
            aa(m):=newAg(j); m:=m+1;
        >>;
    >>;
    aa(m):=0;
    %
    newAg(k):=0; %% to save memory.
    tmp:=mredaa(Ag(k));
    if !-ODE=1 then <<
        if isODE(tmp)=1 then <<
            write "Found target! Answer is FFFF. ";
            ffff:=tmp; !-FFFFFLAG:=1;K:=N;
        >>;
    >>;
    if !-DBG0 neq 0 then
        write "#reduceAg(): M-reduction of Ag(",k,") : "
            ,Ag(k)," --> ",tmp;
        newAg(k):=tmp;
    >>;
>> until (seteqAgAndNewAg()==1 OR !-FFFFFLAG=1);
clear aa,newAg;
end$

% If the set Ag and newAg is equal, then 1 else 0.
procedure seteqAgAndNewAg();
begin
    scalar result,k;
    k:=1;
    while (k>0) do <<
        if Ag(k) neq newAg(k) then <<result:=0; k:=-3;>>
        else <<
            if (Ag(k)=0) and (newAg(k)=0) then <<
                result:=1; k:=-3;
            >>;
        >>;
        k:=k+1;
    >>;
    return result;
end$

% The procedure compute a Grobner basis.
% input: Ag
% output: Ag or FFFF.
procedure gbasis;
begin
    scalar k,j,n,m,tmp,kkk;
    array As(SIZEAS); % work area of G-basis
    !-FFFFFLAG:=0;
    n:=getsizeAg();
    for k:=1:n do As(k):=Ag(k);
    As(n+1):=0; Ag(1):=0;
    repeat <<
        % Ag:=Ag+As
        j:=getsizeAs();
        n:=getsizeAg();

```

/usr/users/z007/basis/Rims/db0.rr

1989/1/14 14:52:2 5 p. 225--

```

write "gbasis(): (reduction of Ag): Size of old Ag is ... ",n;
m:=1;
for k:=(n+1):(n+j+1) do <<Ag(k):=As(m); m:=m+1; >>;
% reduce Ag
reduceAg();

!-FFFFFLAG := checkAgForStop();

if (!-FFFFFLAG = 0) then <<
  n:=getsizeAg();
  write "gbasis():          Compacted. Size of reduced Ag is ... ",n;
  % generate new S-pols
  kkk:=0;As(1):=0;
  K:=0;
  while (K<N) do <<
    K:=K+1;
    J:=K;
    while (J<N) do <<
      J:=J+1;
      tmp:=sp(Ag(k),Ag(j));
      write "gbasis(): S-pol of ",k," and ",j;
      if !-DBG1 neq 0 then
        write "##gbasis(): The candidate is ",tmp;
        tmp:=mredAg(tmp);
        if tmp neq 0 then <<
          kkk:=kkk+1;
          As(kkk):=tmp; As(kkk+1):=0;
          write "gbasis():",kkk
            ,": New s-pol is added. ",tmp;
          if !-ODE=1 then <<
            if isODE(tmp)=1 then <<
              write
                "gbasis(): Found target!",
                " Answer is FFFF. ";
              ffff:=tmp; !-FFFFFLAG:=1;
              K:=N;J:=N;
            >>;
          >>;
        >> else <<
          write
            "gbasis(): The S-pol is reduced to 0.
";
          >>;
        >>;
      >>;
    >> until (getsizeAs() = 0 OR !-FFFFFLAG=1);
    if !-FFFFFLAG neq 1 then <<
      write "gbasis(): Completion is completed.";
      write
        " Basis consists of ",getsizeAg()," elements.(Ag(1),...)";
    >>;
  clear As;
ends$
ends$

```

参考文献

- [Erd1] A. Erdélyi et al., Higher Transcendental Functions. MacGraw-Hill, New York, 1953.
- [Erd2] A. Erdélyi, Transformations of Hypergeometric Functions of Two Variables. Proc. Roy. Soc. Edinburgh Sect A. 62 (1948) 378-385.
- [Fur] A. Furukawa, Gröbner-Bases について, 数理研究記録 (42+1700) 612, 1-23.
- [Hea] A.C. Hearn, REDUCE USER'S MANUAL, Santa Monica, The Rand Corporation, 1988.
- [Oku] K. Okubo, On The Group of Fuchsian Equations, 昭和55年度科研, 研究成果報告書, 1981.
- [Suz-Ono] T. Suzuki, O. Onodera, REDUCEでReduceを, 数理研究記録 663 (1988), 23-39.
- [Tak1] N. Takayama, Gröbner Basis and the Problem of Contiguous Relations, Japan J. Appl. Math., 6 (1989), 147-160.
- [Tak2] N. Takayama, 微分差分作用素用 Gröbner 基底パッケージを用いた超越関数式の零決定問題の解法, 理研プレ予稿, 1989.
- [Zei] D. Zeilberger, A holonomic systems approach to special functions identities, preprint.
- [Nou] M. Noumi, Wronskian determinants and the Gröbner representation of a linear differential equation, preprint, Sophia University.
- [Gal] A. Galligo, Some algorithmic questions on ideals of differential operators. Lect. Note in Computer Sci, 204 (1985), 413-421.